
ORACLE®

Oracle Database 12*c* Release 2 CoreTech Seminar

12.2.0.1
SQL, JSON, ユーティリティ&その他新機能

日本オラクル株式会社
クラウド・テクノロジー事業統括
Database & Exadataプロダクトマネジメント本部
中越祐治

2017/01

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

アジェンダ

1. SQL & PL/SQL
2. JSON
3. ユーティリティ(SQL*Plus, Data Pump, SQL*Loader)
4. グローバリゼーション・サポート
5. SQL & PL/SQL：追加
6. Java & JDBC

アジェンダ

1. SQL & PL/SQL
2. JSON
3. ユーティリティ(SQL*Plus, Data Pump, SQL*Loader)
4. グローバリゼーション・サポート
5. SQL & PL/SQL：追加
6. Java & JDBC

SQLとPL/SQLの主な新機能

- 128バイト識別子のサポート
- TO_BLOB, TO_CHAR, TO_CLOBファンクション
- CONTAINERSヒントの新設

128バイト識別子のサポート(1/2)

NEW IN 12.2

- 識別子長が128バイト
  - 表名、カラム名、ビュー名、パッケージ名、プロシージャ名、etc.
  - compatible >= 12.2.0にて有効
  - compatible <= 12.1.0では、従来通り30バイト
- DBMS_STANDARD.ORA_MAX_NAME_LEN_SUPPORTED新規追加

```
EXEC DBMS_OUTPUT.PUT_LINE(ORA_MAX_NAME_LEN_SUPPORTED);
128
```

128バイト識別子のサポート(2/2)

NEW IN 12.2

35バイト

12.2

```
SQL> create table VERY_VERY_VERY_VERY_LONG_TABLE_NAME
  2   ( VERY_VERY_VERY_VERY_LONG_COLUMN_NAME varchar2(10) not null);

表が作成されました。

SQL> desc VERY_VERY_VERY_VERY_LONG_TABLE_NAME
 名前                                      NULL?    型
----------------------------------------- -------- ----------------------------
VERY_VERY_VERY_VERY_LONG_COLUMN_NAME           NOT NULL VARCHAR2(10)
```

12.2以前

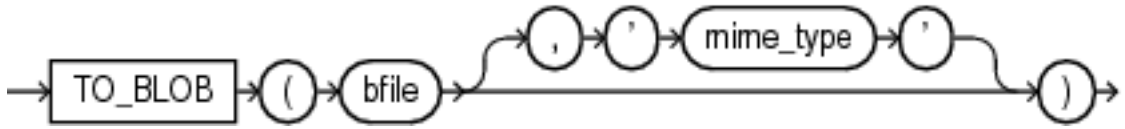
```
SQL> create table VERY_VERY_VERY_VERY_LONG_TABLE_NAME
  2   ( VERY_VERY_VERY_VERY_LONG_COLUMN_NAME varchar2(10) not null);
create table VERY_VERY_VERY_VERY_LONG_TABLE_NAME
             *
行1でエラーが発生しました。:
ORA-00972: 識別子が長すぎます。
```

TO_BLOB, TO_CHAR, TO_CLOBファンクション(1/2)

NEW IN 12.2

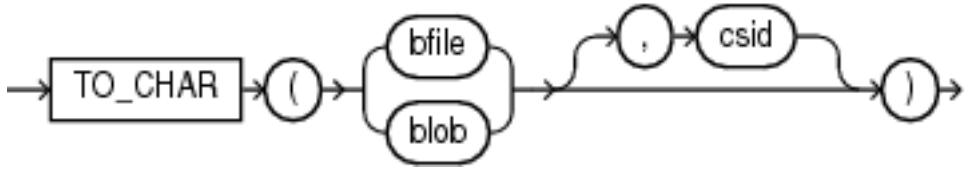
文法

- TO_BLOB: BFILEからBLOBへの変換



JSONデータをBLOBにいれたときに便利

- TO_CHAR: BLOB, BFILEからVARCHAR2への変換



- TO_CLOB: BLOB, BFILEからVARCHAR2への変換

TO_BLOB, TO_CHAR, TO_CLOBファンクション(2/2)

NEW IN 12.2

実行例

- BLOBの出力を読める形式に変換

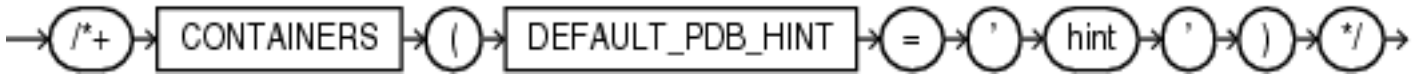
```
SQL> select po_document from j_purchaseorder where id =
'3D7746ADB3122DAAE053D397B90A2909';

PO_DOCUMENT
--------------------------------------------------------------------------------
7B22504F4E756D626572223A393939392C2252656665726E6365223A225048414C4C2D32303134
31303138222C22526571756573746F72223A22506574657220486616C6C222C2255736572223A2250
```

```
SQL> select to_clob(po_document) from j_purchaseorder where id =
'3D7746ADB3122DAAE053D397B90A2909';

TO_CLOB(PO_DOCUMENT)
--------------------------------------------------------------------------------
{"PONumber":9999,"Reference":"PHALL-20141018","Requestor":"P
eter Hall","User":"P
```

CONTAINERSヒントの新設

NEW IN 12.2

- マルチテナント環境での、CDBまたはアプリケーション・コンテナからのCONTAINERS句を使った検索にて、各PDBで実行されるSQLへのヒントを与える。




```
SELECT /*+ CONTAINERS(DEFAULT_PDB_HINT='NO_PARALLEL') */
  (CASE WHEN COUNT(*) < 10000
        THEN 'Less than 10,000'
        ELSE '10,000 or more' END) "Number of Tables"
FROM CONTAINERS(DBA_TABLES);
```

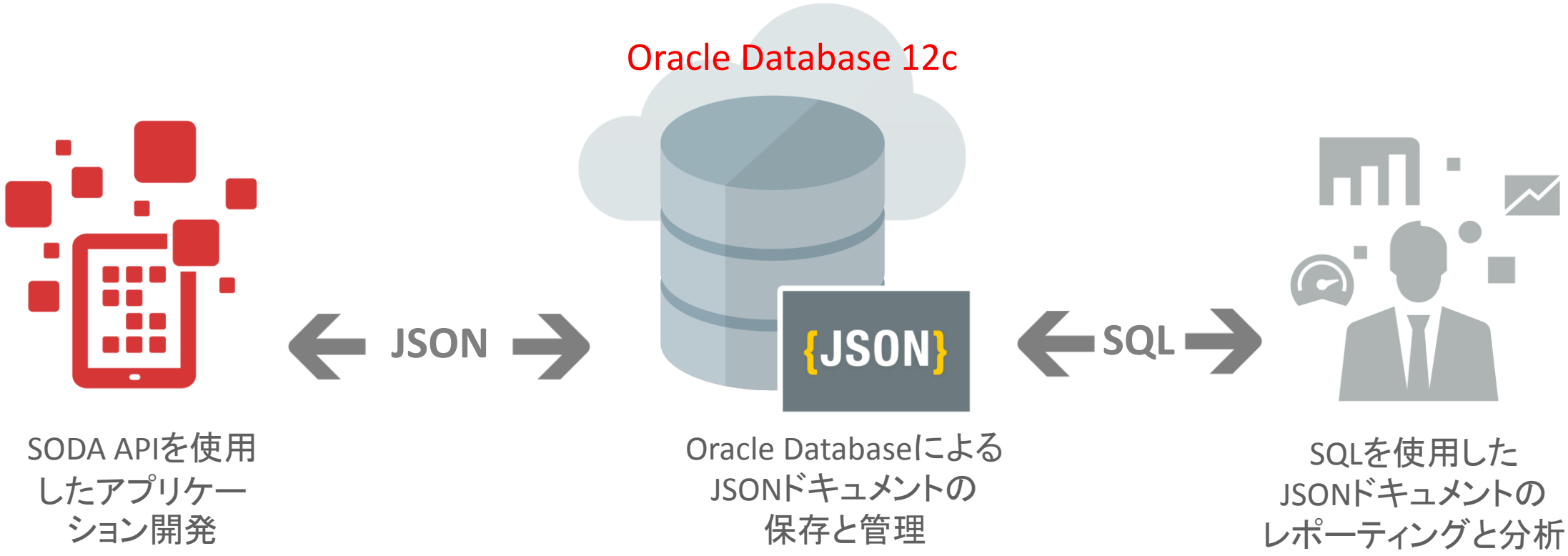
アジェンダ

1. SQL & PL/SQL
2. JSON
3. ユーティリティ(SQL*Plus, Data Pump, SQL*Loader)
4. グローバリゼーション・サポート
5. SQL & PL/SQL : 追加
6. Java & JDBC

ドキュメント・ストアとしてのOracle Database

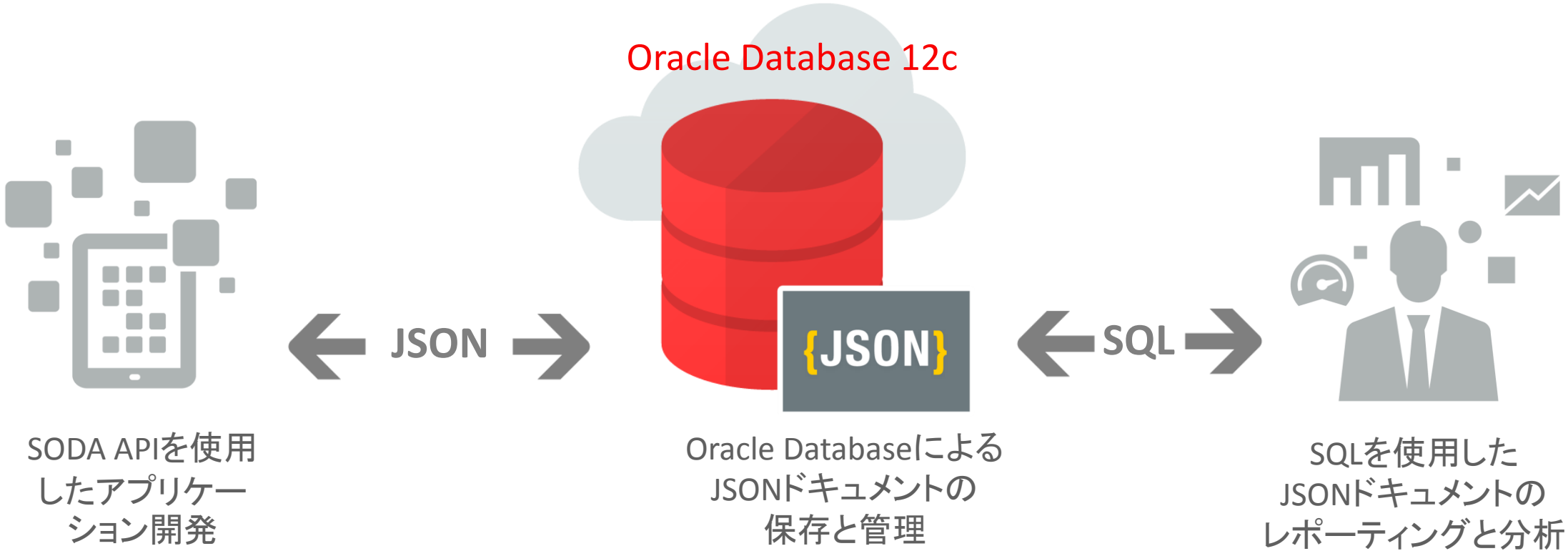
Oracle 12c JSON ドキュメント・ストア

NoSQLでのシンプルな開発



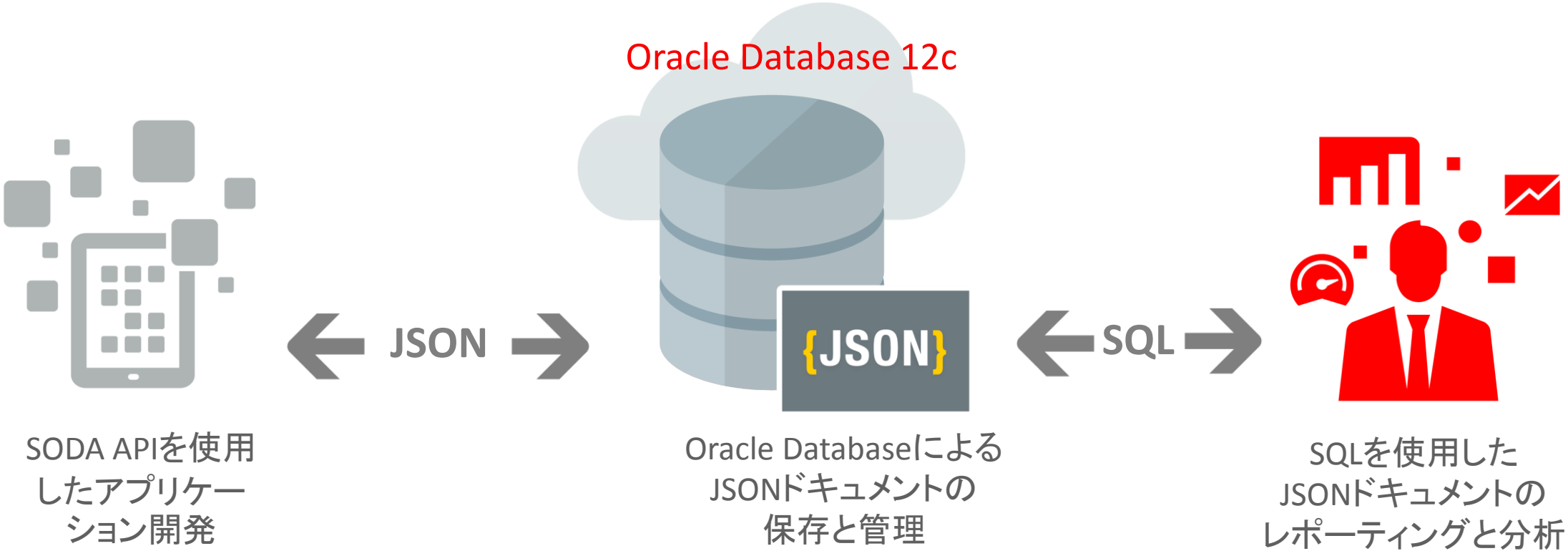
Oracle 12c JSONドキュメント・ストア

エンタープライズ・クラスのデータ管理



Oracle 12c JSON ドキュメント・ストア

SQLのすべての機能が使用可能



独立したマニュアルの提供

独立したマニュアルの提供

NEW IN 12.2

- 12.1まで
  - Oracle XML DB開発者ガイドにふくまれるひとつの章
  - 第Ⅷ部 39 Oracle DatabaseのJSON
- 12.2から
  - JSON Developer Guide
  - 約200ページ
  - http://docs.oracle.com/database/122/ADJSN/toc.htm

JSONカラムのデータ型の推奨

- VARCHAR2(4000), VARCHAR2(32767), BLOB, CLOBのどれかを使用する。
  - JSONは内部でつねにUTF-8として処理を行う。
  - データベースのキャラクタ・セットがAL32UTF8の場合、VARCHAR2にて保存するとキャラクタ・セット変換が発生しない。
  - Exadataでは、一部のJSONの処理がストレージ・セルで行われるのでパフォーマンスが向上する。
- VARCHAR2に収まらない場合はBLOBかCLOB。BLOBが推奨されている。
  - CLOBのエンコーディングはUCS2なので、UTF-8で1バイトの文字を2バイトで保存する。効率が良くない。
  - キャラクタ・セットがAL32UTF8でないデータベースの場合、CLOBで保存するとキャラクタ・セット変換が行われる。
  - 表作成の際に”LOB (column_name) STORE AS (CACHE)”を指定する。
  - SecureFiles LOBを使用する。また、Advanced CompressionオプションによるMedium Compressionを行う。
  - BLOBによる保存の場合to_clob() - 表示, rawtohex() - 保存といった処理を加える必要あり。

JSON関連の主な新機能

JSON関連の主な新機能

- JSONパス式の拡張
- 単純なドット表記法の配列アクセス拡張
- データ・ガイド
- JSONデータを生成する関数
- 新しいPL/SQLオブジェクト・タイプ
- JSON検索索引の機能強化
- JSON_TABLEによるビュー定義
- JSON仮想列をキーとしたパーティション化
- PL/SQL組込みファンクションとしてのJSON関数
- JSON_VALUEとJSON_TABLEでの新しい戻り型対応

JSON検索機能の強化

SQLによるJSONデータの問い合わせ

- 単純な問い合わせ

```
select j.PO_DOCUMENT
from J_PURCHASEORDER j
where j.PO_DOCUMENT.PONumber = 1600
```

- JSONパス式を使用した高度な問い合わせ

```
select JSON_VALUE(PO_DOCUMENT, '$.LineItems[0].Part.UnitPrice' returning NUMBER(5,3))
from J_PURCHASEORDER p
where JSON_VALUE(PO_DOCUMENT, '$.PONumber' returning NUMBER(10)) = 1600
```

  - SQL2017として提案された文法に準拠

JSONパス式の拡張(1/4)

NEW IN 12.2

フィルタ演算子の追加

```
SELECT po_document FROM j_purchaseorder WHERE
 json_exists(po_document, '$?(@.User == "AKHOO" && @.LineItems.Quantity > 8)');
```

(*cond*)	グループ化
cond1 && *cond2*	論理積
cond1 || *cond2*	論理和
! *cond*	否定
exists(@ *path*)	JSONパス式評価

スカラー値の評価
==
!=
<
<=
>=
>

JSONパス式の拡張(2/4)

NEW IN 12.2

PASSING句を使用した変数のバインド, exists句

```
select j.PO_DOCUMENT
 from J_PURCHASEORDER j
 where JSON_EXISTS(
       PO_DOCUMENT,
       '$?(@.PONumber == $PO_NUMBER)'
       passing 1600 as  "PO_NUMBER"
     )
/
```

```
select j.PO_DOCUMENT.PONumber
 from J_PURCHASEORDER j
 where JSON_EXISTS(
       PO_DOCUMENT,
       '$?(@.User == $USER && exists(
                                    @.LineItems?(
                                       @.Part.UPCCode == $UPC
                                       &&
                                       @.Quantity > $QUANTITY
                                    )
                                 )
         )'
       passing 'AKHOO' as "USER", 43396087798 as "UPC", 8 as "QUANTITY"
     )
/
```

- PASSING句により、JSONパス変数としてバインド変数を利用可能
- exists句は配列内のオブジェクトの検索に利用可能

JSONパス式の拡張(3/4)

NEW IN 12.2

アイテム・メソッド

lower()を使う

```
select count(*) from j_purchaseorder
      where json_exists(po_document, '$?(@.Requestor.lower() == "trenna rajs")');
```

upper()を使う

```
select count(*) from j_purchaseorder
      where json_exists(po_document, '$?(@.Requestor.upper() == ”TRENNA RAJS")');
```

length()を使う

```
select count(*) from j_purchaseorder
      where json_exists(po_document, '$?(@.Requestor.length() >= 1)');
```

ORACLE

JSONパス式の拡張(4/4)

JSONアイテム・メソッド一覧

NEW IN 12.2

JSON アイテム・メソッド	
number()	整数への変換。SQLのNUMBER相当。
timestamp()	SQLのDATEへ変換。ISO8601の書式であること。
length()	文字列の長さ。
lower()	小文字に変換。
upper()	大文字に変換。
string()	文字列に変換。
double()	浮動小数への変換。SQLのBINARY_DOUBLE相当。
abs()	絶対値。SQL関数のABS()と同じ動作。
floor()	切り捨て。SQL関数のFLOOR()と同じ動作。
ceiling()	切り上げ。SQL関数のCEIL()と同じ動作。

単純なドット表記法の配列アクセス拡張

NEW IN 12.2

- 以下のような、配列アクセスを含むドット記法に対応

```
select po.po_document.LineItems[*] from j_purchaseorder po;
```
LineItems要素のすべて
```
select po.po_document.LineItems from j_purchaseorder po;
```
と同じ

```
select po.po_document.LineItems[0] from j_purchaseorder po;
```
LineItemsの最初の要素

```
select po.po_document.LineItems[0 to 1].ItemNumber from j_purchaseorder po;
```
１番目と2番目のLineItemsに含まれるItemNumber

JSONドキュメントの解析

データ・ガイド

データ・ガイド(1/7)

JSONドキュメントの解析

NEW IN 12.2

- メタデータの生成: JSONドキュメントの集合から、スキーマ情報を生成する。
  - JSON検索索引について統計を取得すると、最大値、最小値、出現率、スカラー値がJSON nullの数なども取得する。
- 自動作成
  - 仮想列
  - ビュー
    - JSON配列については非正規化を行う。
  - JSONドキュメントの構造をレポート

データ・ガイド(2/7)

NEW IN 12.2

概要

JSONを含む表

```
SQL> desc MOVIE_TICKETS
 Name                                      Type
 ----------------------------------------- ----------------------
 BOOKING_ID                                RAW(16)
 BOOKING_TIME                              TIMESTAMP(6)
 BOOKING_DETAILS                           VARCHAR2(4000)

SQL>
```

dbms_json.
create_view()
create_view_on_path()

ビュー

```
SQL> desc MOVIE_TICKETS_VIEW
 Name                                      Type
 ----------------------------------------- ----------------------
 BOOKING_ID                                RAW(16)
 BOOKING_TIME                              TIMESTAMP(6)
 BOOKING_DETAILS                           VARCHAR2(4000)
 BOOKING_DETAILS$Movie                     VARCHAR2(16)
 BOOKING_DETAILS$Theater                   VARCHAR2(16)
SQL>
```

dbms_json.
add_virtual_columns()

仮想列の追加

```
SQL> desc MOVIE_TICKETS
 Name                                      Type
 ----------------------------------------- ----------------------
 BOOKING_ID                                RAW(16)
 BOOKING_TIME                              TIMESTAMP(6)
 BOOKING_DETAILS                           VARCHAR2(4000)
 BOOKING_DETAILS$Movie                     VARCHAR2(16)
 BOOKING_DETAILS$Theater                   VARCHAR2(16)
SQL>
```

```
{
  “Movie”: “Star Wars”,
  “Theater”: “AMC14”,
  “StartTime”: “2016-03-02T19:30:00”,
  “Tickets”: {
    “Adults”: 2
  }
}
```

データガイド

```
[
{"o:path":"$.Movie","type":"string","o:length":16},
{"o:path":"$.Ticket","type":"object","o:length":16},
{"o:path":"$.Ticket.Adults","type":"number","o:length":1},
{"o:path":"$.Theater","type":"string","o:length":8},
{"o:path":"$.StartTime","type":"string","o:length":32}
]
```

json_dataguide(), json_hierdataguide(), dbms_json.get_index_dataguide()…

データ・ガイド(3/7)

NEW IN 12.2

データ・ガイドの生成

	出力形式 - フラット	出力形式 - 階層
JSON検索索引あり	select dbms_json.get_index_dataguide('表', 'JSONカラム', DBMS_JSON.FORMAT_FLAT) from dual;	select dbms_json.get_index_dataguide('表', 'JSONカラム', DBMS_JSON.FORMAT_HIERARCHICAL) from dual;
JSON検索索引なし	select json_dataguide(JSONカラム) from 表	select json_heirdataguide(JSONカラム) from 表

- データ・ガイドに対応したJSON検索索引があるとJSONデータの更新に追従して、データ・ガイドが更新される。
  - ALL/DBA/USER_JSON_DATAGUIDESから参照可能
- 検索索引が作れない、または効率が悪いときに、索引なしのファンクションを使用する。

データ・ガイド(4/7)

NEW IN 12.2

生成されたデータ・ガイド

フラット

```
[
    {
        "o:path": "$.User",
        "type": "string",
        "o:length": 8
    },
    {
        "o:path": "$.PONumber",
        "type": "number",
        "o:length": 8
    },
    {
        "o:path": "$.LineItems",
        "type": "array",
        "o:length": 1024
    },
    {
        "o:path": "$.LineItems.Part",
        "type": "object",
        "o:length": 256
    },
    {
        "o:path": "$.LineItems.Part.UPCCode",
        "type": "number",
        "o:length": 16
    },
```

階層

```
{
    "type": "object",
    "properties": {
        "User": {
            "type": "string",
            "o:length": 8,
            "o:preferred_column_name": "User"
        },
        "PONumber": {
            "type": "number",
            "o:length": 8,
            "o:preferred_column_name": "PONumber"
        },
        "LineItems": {
            "type": "array",
            "o:length": 1024,
            "o:preferred_column_name": "LineItems",
            "items": {
                "properties": {
                    "Part": {
                        "type": "object",
                        "o:length": 256,
                        "o:preferred_column_name": "Part",
```

データ・ガイド(5/7)

統計を含むデータ・ガイド

NEW IN 12.2

- JSON検索索引の統計を取得する。
  DBMS_STATS.GATHER_INDEX_STATS(

  OWNNAME => ‘スキーマ名’,

  INDNAME => ‘JSON検索索引名’,

  ESTIMATE_PERCENT => 100);
- データ・ガイドを参照する。

SELECT DATAGUIDE FROM
ALL/DBA/USER_JSON_DATAGUIDES

```
[
  {
    "o:path" : "$.User",
    "type" : "string",
    "o:length" : 8,
    "o:preferred_column_name" : "PO_DOCUMENT$U
ser",
    "o:frequency" : 100,
    "o:low_value" : "DLEE",
    "o:high_value" : "VPATABAL",
    "o:num_nulls" : 0,
    "o:last_analyzed" : "2016-09-27T09:54:23"
  },
  {
    "o:path" : "$.PONumber",
    "type" : "number",
    "o:length" : 8,
    "o:preferred_column_name" : "PO_DOCUMENT$PONumber",
    "o:frequency" : 100,
    "o:low_value" : "1",
    "o:high_value" : "10000",
    "o:num_nulls" : 0,
    "o:last_analyzed" : "2016-09-27T09:54:23"
  },
```

データ・ガイド(6/7)

NEW IN 12.2

ビューまたは仮想列の追加

	ビュー	仮想列
データ・ガイドとしてJSON検索索引を参照	DBMS_JSON.CREATE_VIEW_ON_PATH (viewname VARCHAR2, tablename VARCHAR2, jcolname VARCHAR2, path VARCHAR2, frequency NUMBER DEFAULT 0);	DBMS_JSON.ADD_VIRTUAL_COLUMNS (tablename IN VARCHAR2, jcolname IN VARCHAR2, frequency NUMBER DEFAULT 0, hidden BOOLEAN DEFAULT FALSE);
データ・ガイドを引数としてとる (データ・ガイドは階層形式であること)	DBMS_JSON.CREATE_VIEW (viewname VARCHAR2, tablename VARCHAR2, jcolname VARCHAR2, dataguide CLOB);	DBMS_JSON.ADD_VIRTUAL_COLUMNS (tablename IN VARCHAR2, jcolname IN VARCHAR2, dataguide IN CLOB);

データ・ガイド(7/7)

NEW IN 12.2

引数について

- viewname - 作成するビューの名前
- tablename - JSONカラムを含む表名
- jcolname - JSONカラム名
- path – JSONパス式(作成するカラムの絞り込みに使用)
- frequency – JSONドキュメントに出現している割合にたいしてのフィルタ
  - 0は全て。例えば20であれば、全ドキュメントにたいして20%以上含まれる項目。
- hidden - 非表示列とする
- dataguide - データ・ガイド

リレーショナル・データからJSONデータ生成

JSONデータを生成する関数(1/4)

NEW IN 12.2

- SQL標準として定義されたファンクションの使用
  - JSON_ARRAY, JSON_OBJECT, JSON_ARRAYAGGおよびJSON_OBJECTAGG
  - 入れ子でファンクションを呼び出すことにより、複雑なJSONドキュメントの生成が可能
- SQLクエリーによる簡単なJSONドキュメントの生成
  - 文字列として繋ぎあわせることによる、JSON書式の間違いを排除
- パフォーマンスの向上
  - サーバー側でJSONドキュメントを生成することにより、クライアント・サーバー間のラウンドトリップを削減

JSONデータを生成する関数(2/4)

NEW IN 12.2

json_array - 引数を配列の要素として、JSON配列を作成

```
SQL> select JSON_ARRAY(EMPLOYEE_ID, FIRST_NAME, LAST_NAME) JSON
  2    from HR.EMPLOYEES
  3   where EMPLOYEE_ID = 100;

JSON
--------------------------------------------------------------------------------
[ 100 , "Steven" , "King" ]

SQL>
```

- JSON配列は、それぞれの引数から、ひとつづつアイテムを含む。
- 配列には、異なった種類のアイテムを含むことができる。

JSONデータを生成する関数(3/4)

NEW IN 12.2

json_object – キーバリューペアを指定し、JSONオブジェクトを作成

```
SQL> select JSON_OBJECT('Id' is EMPLOYEE_ID, 'FirstName' is FIRST_NAME,
 2                      'LastName' is LAST_NAME) JSON
 3      from HR.EMPLOYEES
 4     where EMPLOYEE_ID = 100;

JSON
--------------------------------------------------------------------------------
{ "Id" : 100 , "FirstName" : "Steven" , "LastName" : "King" }

SQL>
```

- JSONオブジェクトは、それぞれの引数をキー・バリュー・ペアとして含む。

JSONデータを生成する関数(4/4)

NEW IN 12.2

json_arrayagg - 指定した値をひとつの配列として集約

```
select JSON_OBJECT(
         'departmentId' is d.DEPARTMENT_ID,
         'name' is d. DEPARTMENT_NAME,
         'employees' is (
            select JSON_ARRAYAGG(
                     JSON_OBJECT(
                       'employeeId' is EMPLOYEE_ID,
                       'firstName' is FIRST_NAME,
                       'lastName' is LAST_NAME,
                       'emailAddress' is EMAIL
                     )
                   )
              from HR.EMPLOYEES e
             where e.DEPARTMENT_ID = d.DEPARTMENT_ID
            )
       ) DEPT_WITH_EMPLOYEES
  from HR.DEPARTMENTS d
 where DEPARTMENT_NAME = 'Executive';
```

```
DEPT_WITH_EMPLOYEES
----------------------------------------
{
  "departmentId": 90,
  "name": "Executive",
  "employees": [
    {
      "employeeId": 100,
      "firstName": "Steven",
      "lastName": "King",
      "emailAddress": "SKING"
    }, {
      "employeeId": 101,
      "firstName": "Neena",
      "lastName": "Kochhar",
      "emailAddress": "NKOCHHAR"
    }, {
      "employeeId": 102,
      "firstName": "Lex",
      "lastName": "De Haan",
      "emailAddress": "LDEHAAN"
    }
  ]
}
```

- JSON配列を入れ子にしたサブ・クエリーから生成する。

JSONドキュメントの編集

新しいPL/SQLオブジェクト・タイプ(1/3)

NEW IN 12.2

追加されたオブジェクト・タイプ

- スクラッチからJSONオブジェクトを作成する、または、メモリ上でJSONオブジェクトを編集するために使用
- JSON_ELEMENT_T – JSON_OBJECT_T, JSON_ARRAY_T, JSON_SCALAR_Tのスーパータイプ。
- JSON_OBJECT_T
  - JSONオブジェクトのタイプ。オブジェクトの要素の追加、削除はこのタイプのインスタンスに対して実施。
- JSON_ARRAY_T
  - JSON配列のタイプ。配列の要素の追加、削除はこのタイプのインスタンスに対して実施。
- JSON_SCALAR_T
  - JSONのスカラー値を保持。
- JSON_KEY_LIST
  - VARRAY(32767) OF VARCHAR2(4000) – JSON_OBJECT_TのGET_KEYSの返り値の型として使用。

ORACLE

新しいPL/SQLオブジェクト・タイプ(2/3)

NEW IN 12.2

主なメソッド

- parse()
  - JSONデータを保持している変数またはカラムから、JSON_ELEMENT_Tオブジェクトを取り出す。
- is_Array(), is_Object(), is_String() ,etc.
  - キー・値・ペアの、値の型を確認する。
- get, put
  - キー・値・ペアの値を、オブジェクトまたは配列としてアクセスする。
- get_String, get_Number:
  - キー・値・ペアの値を、スカラー値としてアクセスする。
- stringify, to_string
  - PL/SQL JSONデータをテキスト表現に戻す。

新しいPL/SQLオブジェクト・タイプ(3/3)

NEW IN 12.2

オブジェクト・タイプの使用例

実行スクリプト:

```
declare
  je json_element_t;
  jo json_object_t;
begin
  je := json_element_t.parse('{"name":"Radio controlled plane"}');
  if (je.is_object) then
    jo := treat(je as json_object_t);
    jo.put('price', 149.99);
  end if;
  dbms_output.put_line(je.to_string);
end;
/
```

実行結果:

```
{"name":"Radio controlled plane","price":149.99}
```

JSONをパースしてJSON_ELEMENT_Tのインスタンスを生成

JSONオブジェクトとして操作するために、JSON_OBJECT_Tにキャスト

キー・バリュー・ペアとして{“price”:149.99}を追加

JSON問い合わせのパフォーマンス向上

JSON検索索引の機能強化(1/4)

NEW IN 12.2

JSON検索索引を作成するコマンド

- 12.2にて検索索引を作成するコマンドが追加

```
CREATE SEARCH INDEX po_search_idx ON j_purchaseorder (po_document) FOR JSON;
```

- 検索索引の実体はOracle Text索引で12.1と変わらないが、12.1にて作成した検索索引はドロップして、CREATE SEARCH INDEXにて再作成が推奨。
- 数値、日付、時刻について範囲指定が可能
- パーティション表についても、検索索引の作成が可能

JSON検索索引の機能強化(2/4)

索引がない場合の実行計画

- 検索式

```
select count(*) from j_purchaseorder po
        where json_exists(po_document, '$?(@.PONumber > 1000)');
```

- 実行計画 – 表のフルスキャン

```
-------------------------------------------------------------------------------
| Id  | Operation           | Name              | Rows | Bytes    | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                   |    1 |     1994 |   855   (1)| 00:00:01|
|   1 |  SORT AGGREGATE     |                   |    1 |     1994 |            |          |
|*  2 |   TABLE ACCESS FULL| J_PURCHASEORDER | 100  |     194K |   855   (1)| 00:00:01|
-------------------------------------------------------------------------------
```

JSON検索索引の機能強化(3/4)

NEW IN 12.2

JSON検索索引あり

- JSON検索索引の作成

```
CREATE SEARCH INDEX po_search_idx ON j_purchaseorder (po_document) FOR JSON;
```

- 実行計画 – JSON検索索引を使用 - 数値を認識

```
-------------------------------------------------------------------------------
| Id  | Operation            | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                |     1 |  2006 |     4   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE      |                |     1 |  2006 |            |          |
|*  2 |   DOMAIN INDEX       | PO_SEARCH_IDX  |     5 | 10030 |     4   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

JSON検索索引の機能強化(4/4)

JSON_VALUEを使ったファンクション索引あり

- JSON_VALUEを使ったファンクション索引の作成

```
create index po_number_idx on j_purchaseorder(json_value(po_document, '$.PONumber'
returning number error on error));
```

- 実行計画 – ファンクション索引を優先使用

```
-----------------------------------------------------------------------------------
| Id  | Operation           | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                  |     1 |    13 |     2   (0)| 00:00:01|
|   1 |  SORT AGGREGATE     |                  |     1 |    13 |            |          |
|*  2 |   INDEX RANGE SCAN  | PO_NUMBER_IDX    |   500 |  6500 |     2   (0)| 00:00:01|
-----------------------------------------------------------------------------------
```

注) error on errorで作成した索引のみ検索時に使用された。

JSON_TABLEによるビュー定義

マテリアライズド・ビューとしても作成可能

```
CREATE MATERIALIZED VIEW j_purchaseorder_materialized_view
  BUILD IMMEDIATE
  REFRESH FAST ON COMMIT WITH PRIMARY KEY
  AS SELECT po.id, d.*
       FROM j_purchaseorder po,
            json_table(po.po_document, '$'
              COLUMNS (
                po_number         NUMBER(10)          PATH '$.PONumber',
                reference         VARCHAR2(30 CHAR)   PATH '$.Reference',
                requestor         VARCHAR2(128 CHAR)  PATH '$.Requestor',
                userid            VARCHAR2(10 CHAR)   PATH '$.User',
...
```

```
create materialized view log on j_purchaseorder with primary key;
```

その他

JSON仮想列をキーとしたパーティション化

NEW IN 12.2

- パーティション・キーとする仮想列はjson_valueにて取り出す。
- json_valueのRETURNING句によって戻り値の型を指定する。
- json_valueのパスに、フィルターといった条件式 - 述語 -を含まない。
- 仮想列(po_num_vc)の元となるJSONカラム(po_document)はis jsonが真であることが条件。
- データ1行挿入時にjson_valueが呼ばれるのは処理として重いので、できれば仮想列は使わず、通常の列をパーティション・キーにする方を推奨。

```
CREATE TABLE j_purchaseorder_partitioned
  (id VARCHAR2 (32) NOT NULL PRIMARY KEY,
   date_loaded TIMESTAMP (6) WITH TIME ZONE,
   po_document BLOB
   CONSTRAINT ensure_json_part CHECK (po_document IS JSON),
   po_num_vc NUMBER GENERATED ALWAYS AS
     (json_value (po_document, '$.PONumber' RETURNING NUMBER)))
  LOB (po_document) STORE AS (CACHE)
  PARTITION BY RANGE (po_num_vc)
   (PARTITION p1 VALUES LESS THAN (1000),
    PARTITION p2 VALUES LESS THAN (2000),
    PARTITION p3 VALUES LESS THAN (3000)
);
```

以前のバージョンではエラー
ORA-14135: a LOB column cannot serve as a partitioning column

PL/SQL組込みファンクションとしてのJSON関数

NEW IN 12.2

- json_value, json_query, json_object, json_array, json_existsがPL/SQLのビルトイン関数として利用可能
- 以下のような記述に変更できる。

```
declare
js varchar2(4000);
begin
  select json_query('{"doc":["a","b","c"]}','$.doc[0]' with wrapper) into js from dual;
  dbms_output.put_line(js);
end;
/
```

```
declare
js varchar2(4000);
begin
  js := json_query('{"doc":["a","b","c"]}','$.doc[0]' with wrapper);
  dbms_output.put_line(js);
end;
/
```

JSON_VALUEとJSON_TABLEでの新しい戻り型対応

NEW IN 12.2

- DATE, TIMESTAMP, TIMESTAMP with TIMEZONEが追加

- 日付の指定例
  - json_value('{"t":"20160909"}','$.t' returning date error on error);
- 時刻の指定例
  - json_value('{"t":"20160909125900"}','$.t' returning timestamp error on error);
- タイムゾーン付き時刻の指定例
  - json_value('{"t":"20160909125900+0900"}','$.t' returning timestamp with time zone error on error);

アジェンダ

1. SQL & PL/SQL
2. JSON
3. ユーティリティ(SQL*Plus, Data Pump, SQL*Loader)
4. グローバリゼーション・サポート
5. SQL & PL/SQL：追加
6. Java & JDBC

SQL*Plus

SQL*Plusの主な新機能

- コマンド・ヒストリ
- 128バイト識別子のサポート
- 変数へのinput bindingのサポート
- SET MARKUPでのCSV出力のサポート
- SET FEEDBACKへのONLY(行数出力のみ)の追加

コマンド・ヒストリ(1/3)

NEW IN 12.2

新設コマンド

- splplusにコマンド履歴の実行・編集機能が追加
  - SET HIST[ORY] {ON | OFF | n}
    - ヒストリ機能の開始(ON)、停止(OFF)、取得行数の指定 (ON では 100行となる)
    - 起動直後(デフォルト)では無効。
  - HIST[ORY] [n RUN | EDIT | DEL[ELE]] | [CLEAR | LIST]
    - ヒストリからの選択実行(RUN)、編集(EDIT)、削除(DELETE)。
    - ヒストリの初期化(CLEAR)と一覧表示(LIST)。
    - デフォルトはLIST。
  - DEFINE _EDITOR = vi
    - 起動するエディターの指定。

コマンド・ヒストリ(2/3)

NEW IN 12.2

EDIT実行例

1
```
SQL> set history on
SQL> show history
historyはONであり、設定先は "100"です。
```

2
```
SQL> history
1   show history
2   show user
3   select count(*) from user_tables;
4   select count(*) from user_objects where
    object_type = 'TABLE';
```

3
```
SQL> history 4 edit
```

エディタの起動

4
```
select count(*) from user_objects where
object_type = 'TABLE';
~
~
~
history.buf 1L,63C
```

エディタの終了

5
```
select count(*) from user_objects where
object_type = 'VIEW';
~
~
~
:wq
```

6
```
SQL> history
1   show history
2   show user
3   select count(*) from user_tables;
4   select count(*) from user_objects where
object_type = 'TABLE';
5 select count(*) from user_objects where
object_type = 'VIEW';
```

末尾に追加

コマンド・ヒストリ(3/3)

NEW IN 12.2

RUN実行例

1

```
SQL> history
1   show history
2   show user
3   select count(*) from user_tables;
4   select count(*) from user_objects where
object_type = 'TABLE';
5 select count(*) from user_objects where
object_type = 'VIEW';
```

履歴からの実行

2

```
SQL> history 5 run

 COUNT(*)
----------
      6429
```

3

```
SQL> history
1   show history
2   show user
3   select count(*) from user_tables;
4   select count(*) from user_objects where
object_type = 'TABLE';
* 5 select count(*) from user_objects where
object_type = 'VIEW';
```

先頭にアスタリスクが付加

128バイト識別子のサポート

NEW IN 12.2

- SQL*Plusの識別子長として30byteから128byteへ拡張

38byte

```
DEFINE very_very_very_long_identifier_support=supported
PROMPT &very_very_very_long_identifier_support
```

- 12.2の結果

```
SQL> PROMPT &very_very_very_long_identifier_support
supported
```

- 12.1の結果

```
SQL> DEFINE very_very_very_long_identifier_support=supported
SP2-0318: "very_..."で始まる記号名が長すぎます(最大 30)。
```

ORACLE

変数へのinput bindingのサポート

NEW IN 12.2

- 以下の記法にて、変数への値の割り当てが可能

```
SQL> variable tmp_var varchar2(10)=Smith;
SQL> exec dbms_output.put_line(:tmp_var);
Smith

PL/SQLプロシージャが正常に完了しました。
```

宣言時にSmithの割り当て

宣言後のJohnの割り当て

```
SQL> variable tmp_var varchar2(10);
SQL> variable tmp_var=John
SQL> exec dbms_output.put_line(:tmp_var);
John

PL/SQLプロシージャが正常に完了しました。
```

SET MARKUPでのCSV出力のサポート

NEW IN 12.2

- SET MARKUP CSV ON [DELIMI[TER] character] [QUOTE {ON|OFF}]
  - DELIMITERによって区切り文字を指定
  - QUOTEのON, OFFによって文字列出力がダブルクォート(“ “)で囲まれる。
    - 文字列にダブルクォートが含まれる場合はエスケープ(“が””として出力)される。

```
SQL> set heading on
SQL> set markup csv on delimiter , quote on
SQL> spool session.csv
SQL> select * from v$session;

"SADDR","SID","SERIAL#","AUDSID","PADDR","USER#","USERNAME","COMMAND","OWNERID",
"TADDR","LOCKWAIT","STATUS","SERVER","SCHEMA#","SCHEMANAME","OSUSER","PROCESS","
MACHINE","PORT","TERMINAL","PROGRAM","TYPE","SQL_ADDRESS","SQL_HASH_VALUE","SQL_
ID","SQL_CHILD_NUMBER","SQL_EXEC_START","SQL_EXEC_ID","PREV_SQL_ADDR","PREV_HASH
_VALUE","PREV_SQL_ID","PREV_CHILD_NUMBER","PREV_EXEC_START","PREV_EXEC_ID","PLSQ
L_ENTRY_OBJECT_ID","PLSQL_ENTRY_SUBPROGRAM_ID","PLSQL_OBJECT_ID","PLSQL_SUBPROGR
AM_ID","MODULE","MODULE_HASH","ACTION","ACTION_HASH","CLIENT_INFO","FIXED_TABLE_
SEQUENCE","ROW_WAIT_OBJ#","ROW_WAIT_FILE#","ROW_WAIT_BLOCK#","ROW_WAIT_ROW#","TO
P_LEVEL_CALL#","LOGON_TIME","LAST_CALL_ET","PDML_ENABLED","FAILOVER_TYPE","FAILO
VER_METHOD
```

ヘッダー行の制御はheadingを使う
ファイル出力にはspoolを使う
区切り文字を”,”とし、クオートする指定

CSV形式の出力

SET FEEDBACKへのONLY(行数出力のみ)の追加

NEW IN 12.2

- SET FEED[BACK] {6 | n | ON | OFF | ONLY}
  - ONLYが追加
  - クエリー時に行数のみを表示する。

```
SQL> set feedback only
SQL> select * from v$session;

49行が選択されました。

SQL> show feedback
feedback ONLY
SQL>
```

Data Pump Export/Import

Data Pumpの主な新機能

- インポートおよびエクスポートの並列処理拡張
- ファイル名の置換変数の追加
- TRANSPORT_DATAFILESでのワイルドカード指定の追加
- REMAP_DIRECTORYオプションの追加

インポートおよびエクスポートの並列処理拡張

NEW IN 12.2

- expdpおよびimpdpのPARALLEL指定により、複数ワーカー・プロセスによる並列処理が行われる。
  - 12.2からメタデータも並列処理の対象。
  - インポート時、従属性のあるオブジェクトについては、特定のワーカー・プロセスでシリアル処理

```
impdp hr DIRECTORY=dpump_dir1 LOGFILE=parallel_import.log
JOB_NAME=imp_par3 DUMPFILE=par_exp%U.dmp PARALLEL=3
```

```
expdp hr DIRECTORY=dpump_dir1 LOGFILE=parallel_export.log
JOB_NAME=par4_job DUMPFILE=par_exp%u.dmp PARALLEL=4
```

ファイル名の置換変数の追加(1/2)

NEW IN 12.2

置換変数一覧

置換変数	置換結果
%U	既設。01から99までの数字2文字に置き換え。expdp, impdp双方で指定可能。
%l, %L	新設。%Uは1から99までだが、これを1から2147483646まで対応したもの。99より大きいと3桁、999より大きいと4桁と桁数が増える。expdp, impdp双方で指定可能。
%d, %D	新設。日付に置き換わる。DD。９月７日であれば、07。expdpのみ。
%m, %M	新設。月に置き換わる。MM。９月７日であれば、09。expdpのみ。
%y, %Y	新設。年に置き換わる。YYYY。expdpのみ。
%t, %T	新設。年月日に置き換わる。YYYYMMDD。expdpのみ。

ファイル名の置換変数の追加(2/2)

NEW IN 12.2

置換変数の使用例

```
$ expdp dpuser/dpuser@localhost/dpdb DIRECTORY=dpump_dir1 LOGFILE=parallel_export.log
JOB_NAME=par4_job DUMPFILE=par_exp%u-%t.dmp PARALLEL=4

Export: Release 12.2.0.1.0 - Production on 金 9月 23 14:36:47 2016

Copyright (c) 1982, 2016, Oracle and/or its affiliates.  All rights reserved.

接続先: Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
"DPUSER"."PAR4_JOB"を起動しています: dpuser/********@localhost/dpdb DIRECTORY=dpump_dir1
LOGFILE=parallel_export.log JOB_NAME=par4_job DUMPFILE=par_exp%u-%t.dmp PARALLEL=4
オブジェクト型SCHEMA_EXPORT/TABLE/TABLE_DATAの処理中です
[中略]
オブジェクト型SCHEMA_EXPORT/TABLE/TABLEの処理中です
マスター表"DPUSER"."PAR4_JOB"は正常にロード/アンロードされました
******************************************************************************
DPUSER.PAR4_JOBに設定されたダンプ・ファイルは次のとおりです:
  /home/oracle/testcase/0050_devtools/004_datapump/dpump_dir1/par_exp01-20160923.dmp
  /home/oracle/testcase/0050_devtools/004_datapump/dpump_dir1/par_exp02-20160923.dmp
  /home/oracle/testcase/0050_devtools/004_datapump/dpump_dir1/par_exp03-20160923.dmp
  /home/oracle/testcase/0050_devtools/004_datapump/dpump_dir1/par_exp04-20160923.dmp
ジョブ"DPUSER"."PAR4_JOB"が金 9月 23 14:37:19 2016 elapsed 0 00:00:30で正常に完了しました
```

TRANSPORT_DATAFILESでのワイルドカード指定の追加

NEW IN 12.2

トランスポータブル表領域モードによるexpdp

```
expdp system/manager@localhost/dpdb DIRECTORY=dpump_dir1 LOGFILE=tbs_export.log
JOB_NAME=tbs_job DUMPFILE=tbs_exp%u.dmp transport_tablespaces=testtbs,users
```

処理後にデータファイルのコピーを求められる:

Datafiles required for transportable tablespace TESTTBS:
/u01/app/oracle/oradata/CDB122/3BE6D38BE543266FE053AD97B90AB2D7/datafile/o1_mf_testtbs_cwzjqg4c_.dbf
Datafiles required for transportable tablespace USERS:
/u01/app/oracle/oradata/CDB122/3BE6D38BE543266FE053AD97B90AB2D7/datafile/o1_mf_users_cwzj8oz6_.dbf

なので

cp /u01/app/…/datafile/o1_mf_testtbs_cwzjqg4c_.dbf /tmp/testtbs.dbf
cp /u01/app/…/datafile/o1_mf_users_cwzj8oz6_.dbf /tmp/users.dbf

*と?が使用可能

トランスポータブル表領域モードによるimpdp

```
impdp system/manager@localhost/dpdb DIRECTORY=dpump_dir1 LOGFILE=tbs_imp.log
JOB_NAME=tbs_imp_job DUMPFILE=tbs_exp%u.dmp transport_datafiles='/tmp/*.dbf'
```

REMAP_DIRECTORYオプションの追加

NEW IN 12.2

- CREATE TABLESPACE, CREATE LIBRARY, CREATE DIRECTORYを対象として、左側のディレクトリ・パスを置き換える。
- ディレクトリを変更する複数のREMAP_DATAFILEをひとつのREMAP_DIRECTORYに置き換えられる。

```
REMAP_DATAFILE="/u01/app/oradata/CDB1/system.dbf':'/u02/app/oracle/oradata/CDB2/system.dbf'"
REMAP_DATAFILE="/u01/app/oradata/CDB1/sysaux.dbf':'/u02/app/oracle/oradata/CDB2/sysaux.dbf'"
REMAP_DATAFILE="/u01/app/oradata/CDB1/users.dbf':'/u02/app/oracle/oradata/CDB2/users.dbf'"
```

```
REMAP_DIRECTORY="/u01/app/oradata/CDB1/':'/u02/app/oracle/oradata/CDB2/'"
```

ORACLE

SQL*Loader

SQL*Loaderの主な新機能

- SDF_PREFIXオプションの追加
- エクスプレス・モードの文字列によるパラメータ指定
- EMPTY_LOBの取り扱いの指定
- カラムのデフォルトの扱いを変更するDEFAULTSオプション
- 一度に評価するデフォルト値の数
- 表のロック待機の動作制御

SDF_PREFIXオプションの追加

NEW IN 12.2

- セカンダリ・データ・ファイル(SDF)の位置を指定するオプション

```
sqlldr userid=ldruser/ldruser@localhost/ldrpdb control=picts.ctl log=picts.log sdf_prefix=lobdir/
```

picts.ctl - コントロール・ファイル

```
LOAD DATA
INFILE 'sample.dat'
   INTO TABLE person_table
   FIELDS TERMINATED BY ','
   (name        CHAR(20),
   ext_fname     FILLER CHAR(40),
   "RESUME"      LOBFILE(ext_fname) TERMINATED BY EOF)
```

プライマリ・データ・ファイル

```
Johny Quest,jqresume.txt,
Speed Racer,srresume.txt,
```

SDF_PREFIX付加

セカンダリ・データ・ファイル

```
lobdir/jqresume.txt
lobdir/srresume.txt
```

エクスプレス・モードの文字列によるパラメータ指定

NEW IN 12.2

- エクスプレス・モードでのENCLOSED_BY, OPTIONALLY_ENCLOSED_BY, TERMINATED_BYの指定が1文字から文字列に拡張

```
sqlldr userid=ldruser/ldruser@localhost/ldrpdb table=simple_value optionally_enclosed_by='<>'
```

```
simple_value.dat
<>abc<>
<>def<>
<>ghi<>
```

以前のバージョンでは:
SQL*Loader-186: Invalid value for the OPTIONALLY_ENCLOSED_BY parameter.
SQL*Loader-301: string for TERMINATED BY or ENCLOSED BY clause is longer than 1 bytes

EMPTY_LOBの取り扱いの指定

NEW IN 12.2

- EMPTY_LOBS_ARE_NULL = { TRUE | FALSE }
  - TRUEの場合、空のLOBをNULLに設定する。
  - FALSEの場合、EMPTY_LOB()をLOBに設定する。

```
create table t
(
   c0 varchar2(10),
   c1 clob
) ;
```

```
options (empty_lobs_are_null=true)
load data
infile *
truncate
into table t
fields terminated by ','
trailing nullcols
(
  c0 char,
  c1 char
)
begindata
1,,
```

true

```
set null '<NULL>'
select * from t where c1
is null;
C0         C1
---------- -----------
1          <NULL>
```

C1はNULL

false

```
set null '<NULL>'
select * from t where c1
is not null;
C0         C1
---------- -----------
1          <NULL>
```

C1は非NULLだが、空文字列なので表示は'<NULL>'

カラムのデフォルトの扱いを変更するDEFAULTS(1/3)

NEW IN 12.2

指定方法

- DEFAULTS={IGNORE | IGNORE_UNSUPPORTED_EVALUATE_ONCE | IGNORE_UNSUPPORTED_EVALUATE_EVERY_ROW | EVALUATE_ONCE | EVALUATE_EVERY_ROW}
  - IGNORE - カラムのDefault指定は無視
  - IGNORE_UNSUPPORTED_EVALUATE_ONCE - 評価は最初の一回。サポートされていないDefault指定は無視。
  - IGNORE_UNSUPPORTED_EVALUATE_EVERY_ROW - 評価は毎行。サポートされていないDefault指定は無視。
  - EVALUATE_ONCE -評価は最初の一回。サポートされていないDefault指定を報告する。シーケンスの場合を除くデフォルト。
  - EVALUATE_EVERY_ROW - 評価は毎行。サポートされていないDefault指定を報告する。シーケンスの場合はデフォルト。

カラムのデフォルトの扱いを変更するDEFAULTS(2/3)

NEW IN 12.2

実行例

```
create sequence test_s;
create table test
(
   c0 varchar2(10),
   c1 number default test_s.nextval
);
```

```
load data
DEFAULT EXPRESSION CACHE 1000
infile *
append
into table test
fields terminated by ','
trailing nullcols
(
  c0 char
)
begindata
1,
2,
```

defaults=ignore

```
C0                   C1
-------------------- ----------
1                    <NULL>
2                    <NULL>
1                    <NULL>
2                    <NULL>
```

defaults=evaluate_once

```
C0                   C1
-------------------- ----------
1                    1
2                    1
1                    2
2                    2
```

defaults=evaluate_every_row

```
C0                   C1
-------------------- ----------
1                    1
2                    2
1                    1001
2                    1002
```

カラムのデフォルトの扱いを変更するDEFAULTS(3/3)

NEW IN 12.2

一度に評価するデフォルト値の数

- DEFAULT EXPRESSION CACHE *n*
  - コントロール・ファイルに記述
- カラムのDEFAULT指定がシーケンスを含んだ場合、ダイレクト・パス・ロードがコンベンショナル・ロードより遅くなる。
- DEFAULT値の評価を一度に行う数を指定する。デフォルトは100。
  - EVALUATE_EVERY_ROWの場合のみ有効(EVALUATE_ONCEでは最初の1度しか評価しないため)。
- キャッシュ分シーケンスは取得されるため、ロードに使われないと、そのシーケンスは再利用されない。

表のロック待機の動作制御

NEW IN 12.2

- DIRECT_PATH_LOCK_WAIT = { TRUE | FALSE }
  - ダイレクト・パス・モードへのオプション
  - TRUE - 表のロックが取得できるまでsqlldrは待機する。
  - FALSE - ロック取得試行を1秒単位で繰り返す。最大30回実行し、最終的に取得に失敗すると ORA-54 resource busy and acquire with NOWAIT specified or timeout expired を返す。

```
$ sqlldr ldruser/ldruser@localhost/ldrpdb test.ctl direct=true direct_path_lock_wait=false

SQL*Loader: Release 12.2.0.1.0 - Production on Fri Sep 23 17:27:55 2016

Copyright (c) 1982, 2016, Oracle and/or its affiliates.  All rights reserved.

Path used:      Direct
SQL*Loader-951: Error calling once/load initialization
ORA-00604: error occurred at recursive SQL level 1
ORA-00054: resource busy and acquire with NOWAIT specified or timeout expired
```

アジェンダ

1. SQL & PL/SQL
2. JSON
3. ユーティリティ(SQL*Plus, Data Pump, SQL*Loader)
4. グローバリゼーション・サポート
5. SQL & PL/SQL：追加
6. Java & JDBC

グローバリゼーション・サポートの主な新機能

- カラム毎の照合順序指定
- 照合順序のデフォルト
- 照合順序の使用に当たって

カラム毎の照合順序指定

NEW IN 12.2

- COLLATE句による指定
- 日本で使うのは
  - BINARY
  - JAPANESE_M
  - JAPANESE_M_CI (大文字小文字無視)
  - USING_NLS_COMP (デフォルト)

```
create table ling_test
(oid      number,
 k        varchar2(80),
 k_b      varchar2(80) collate binary,
 k_j      varchar2(80) collate japanese_m
);
```

カラム毎の照合順序指定

	order by k (NLS_COMP=ANSI, NLS_SORT=JAPANESE_M)	order by k_b	order by k_j
AL32UTF8	JAPANSE_M	AL32UTF8の昇順	JAPANESE_M
JA16SJIS	JAPANESE_M	JAPANESE_M	JAPANESE_M
JA16EUC	JAPANESE_M	JAPANESE_M	JAPANESE_M

照合順序のデフォルト

NEW IN 12.2

- 表単位でのデフォルト指定
  - CREATE TABLE <TABLE_NAME> (.....) DEFAULT COLLATION *collation_name* ;
  - ALTER TABLE <TABLE_NAME> DEFAULT COLLATION *collation_name* ;
- スキーマ単位でのデフォルト指定
  - CREATE USER <USERNAME> .... DEFAULT COLLATION *collation_name*;
  - ALTER USER <USERNAME> .... DEFAULT COLLATION *collation_name*;
- セッション単位のデフォルト指定
  - ALTER SESSION SET DEFAULT_COLLATION=*collation_name*;
  - スキーマ単位のデフォルト指定を置き換える。
  - ALTER SESSION SET DEFAULT_COLLATION=NONE; - 解除

照合順序の使用に当たって

NEW IN 12.2

- MAX_STRING_SIZE = EXTENDED
  - VARCHAR2の文字数制限を4000から32Kに変更する必要があります。
  - MAX_STRING_SIZEはPDB単位の設定ですが、CDBと一致している必要あり。
    - 異なっていると” ORA-14694: database must in UPGRADE mode to begin MAX_STRING_SIZE migration“として、PDBがオープンしない。
- COMPATIBLE >= 12.2
  - 12.2からの新機能なので、COMPATIBLEパラメータを12.2以上にする必要があります。

MAX_STRING_SIZEの変更手順例(CDBとPDB毎の実施)

```
> alter system set max_string_size = extended scope=spfile;
> shutdown immediate
> startup upgrade
> @?/rdbms/admin/utl32k.sql
> shutdown immediate
> startup
```

ORACLE

アジェンダ

1. SQL & PL/SQL
2. JSON
3. ユーティリティ(SQL*Plus, Data Pump, SQL*Loader)
4. グローバリゼーション・サポート
5. SQL & PL/SQL：追加
6. Java & JDBC

SQLとPL/SQLの新機能:追加

- リテラル指定のかわりに静的PL/SQL式を指定可能
- サブプログラム単位のアクセス制限
- deprecateプラグマの追加
- PL/SQLでのみ使用可能な型のDBMS_SQLでのバインド
- CAST失敗時のデフォルト値指定
- VALIDATE_CONVERSIONファンクションの追加
- LISTAGGファンクションの機能拡張
- 新しいコード・カバレッジ・ツールの提供
- PL/Scopeの拡張

リテラル指定のかわりに静的PL/SQL式を指定可能(1/2)

NEW IN 12.2

識別子長の指定

```
…
   One_Part_ID varchar2(ORA_MAX_NAME_LEN + 2);
   Two_Part_ID varchar2(2 * (ORA_MAX_NAME_LEN + 2) + 1);
begin
   One_Part_ID := '"My Table"';
   Two_Part_ID := '"My Schema"' || '.' || One_Part_ID;
…
```

- 一般に、表名などの識別子を保持する変数としてVARCHAR2(30), VARCHAR2(32)を定義することがある。
- 12.2から識別子長が128バイトに拡張された。
- リテラルの代わりに静的なPL/SQL式を使えるようになったので、修正の際は定数と式を使いましょう。

リテラル指定のかわりに静的PL/SQL式を指定可能(2/2)

NEW IN 12.2

追加されたサブタイプ

- 以下の記述が可能

```
declare
    l  constant number := 80;
    s  varchar2(l);
begin
    …
```

12.2以前は
PLS-00491: numeric literal required

- サブタイプDBMS_ID, DBMS_QUOTED_IDの利用

```
declare
   table_name   dbms_id;          -- varchar2(ora_max_name_len)
   qtable_name  dbms_quoted_id;   -- varchar2(ora_max_name_len + 2)
begin
   …
```

サブプログラム単位のアクセス制限(1/2)

NEW IN 12.2

ACCESSIBLE BYによる制御

- パッケージ仕様部にて、サブプログラムのホワイトリスト定義

```
create or replace package p auth definer
is
  function a return varchar2 deterministic accessible by (x);
  procedure b;
end;
```

- プロシージャaからの呼出し

```
create or replace package body q
is
  procedure a is
  begin
    dbms_output.put_line(p.a());
  end;
end q;
```

コンパイル時エラー発生：PLS-00904: insufficient privilege to access object A

サブプログラム単位のアクセス制限(2/2)

NEW IN 12.2

12.1でのパッケージ単位のアクセス制限

- パッケージ仕様部にて、サブプログラムのホワイトリスト定義

```
create or replace package p auth definer
  accessible by (x)
is
  function a return varchar2 deterministic;
  procedure b;
end;
```

- プロシージャaからの呼出し

```
create or replace package body q
is
  procedure a is
  begin
    dbms_output.put_line(p.a());
  end;
end q;
```

コンパイル時エラー発生：PLS-00904: insufficient privilege to access object P

deprecateプラグマの追加(1/2)

NEW IN 12.2

コンパイル時の警告

```
create or replace procedure p authid definer is
  pragma deprecate(p, 'p is deprecated. You must use p2 instead.');
begin
  dbms_output.put_line('p');
end p;
```

コンパイル時警告発生 : PLW-06019: entity P is deprecated

```
create or replace procedure q authid definer is
begin
  p();
  dbms_output.put_line('q');
end q;
```

コンパイル時警告発生 : PLW-06020: reference to a deprecated entity: P declared in unit P[1,11]. p is deprecated. You must use p2 instead.

deprecateプラグマの追加(2/2)

NEW IN 12.2

実行時の動作

- あくまでもコンパイル時の警告 - 実行時に警告は出力されない。

```
SQL> set serveroutput on
SQL> exec q
p
q

PL/SQL procedure successfully completed.
```

- 警告の出力レベルの設定

```
exec dbms_warning.set_warning_setting_string('ENABLE:INFORMATIONAL','SESSION');
```

ORACLE

PL/SQLでのみ使用可能な型のDBMS_SQLでのバインド(1/3)

NEW IN 12.2

BOOLEANの追加

- BOOLEANのバインド - 新設された呼出し

```
dbms_sql.bind_variable(c in number(38), name in varchar2, value in boolean);
```

- 次のプロシージャが定義されている

```
procedure p(i in boolean) is...
```

- 以下の呼出しが可能

```
  ...
   x constant boolean not null := true;
begin
  DBMS_Sql.Parse(Cur, 'call p(:b)', DBMS_Sql.Native);
  DBMS_Sql.Bind_Variable(Cur, 'b', x);
  Dummy := DBMS_Sql.Execute(Cur);
  DBMS_Sql.Close_Cursor(Cur);
  ...
```

PL/SQLでのみ使用可能な型のDBMS_SQLでのバインド(2/3)

NEW IN 12.2

DBMS_SQL.BIND_VARIABLE_PKGの追加

- レコード型のバインド - 新設された呼出し

```
dbms_sql.bind_variable_pkg(c in number(38), name in varchar2,
                           value in <datatype>);
```

RECORD
VARRAY
NESTED TABLE
INDEX BY PLS_INTEGER TABLE
INDEX BY BINARY_INTEGER TABLE
(VARARY, NESTED_TABLEはdbms_sql.bind_variableにある)

PL/SQLでのみ使用可能な型のDBMS_SQLでのバインド(3/3)

NEW IN 12.2

RECORD型のバインド例

- 次のプロシージャが定義されている

```
package Pkg authid Definer is
  type r is record(a integer, b integer);
  procedure p(i in r);
end Pkg;
```

- 以下の呼出しが可能

```
 ...
  x Pkg.r;
begin
  x.a := 17;
  x.b := 42;
  DBMS_Sql.Parse(Cur, 'call Pkg.p(:b)', DBMS_Sql.Native);
  DBMS_Sql.Bind_Variable_Pkg(Cur, 'b', x);
  Dummy := DBMS_Sql.Execute(Cur);
  DBMS_Sql.Close_Cursor(Cur);
```

CAST失敗時のデフォルト値指定(1/2)

NEW IN 12.2

指定方法

- 追加されたDEFAULT指定

```
CAST({ expr | MULTISET (subquery) } AS type_name [ DEFAULT return_value ON
CONVERSION ERROR ] [, fmt [, 'nlsparam' ] ])
```

- CASTと同等のファンクションにも追加
  - TO_BINARY_DOUBLE, TO_BINARY_FLOAT, TO_DATE, TO_DSINTERVAL, TO_NUMBER, TO_TIMESTAMP, TO_TIMESTAMP_TZ, TO_YMINTERVAL
- TO_DATEでは以下

```
TO_DATE(char [ DEFAULT return_value ON CONVERSION ERROR ] [, fmt [,
'nlsparam' ] ])
```

ORACLE

CAST失敗時のデフォルト値指定(2/2)

NEW IN 12.2

実行例

- 'N/A'はNUMBERに変換できないので、0を返す。

```
SELECT CAST('N/A'
  AS NUMBER
  DEFAULT '0' ON CONVERSION ERROR) AS V
FROM DUAL;

         V
----------
         0
```

- エラーが発生したら、2000年1月1日午前1時を返す。

```
SELECT CAST('1999-12-01 11:00:00 +9:00'
  AS TIMESTAMP WITH TIME ZONE
  DEFAULT '2000-01-01 01:00:00 +9:00'
  ON CONVERSION ERROR,
  'YYYY-MM-DD HH:MI:SS TZH:TZM',
  'NLS_DATE_LANGUAGE = American') AS V
FROM DUAL;

V
---------------------------------------
01-DEC-99 11.00.00.000000000 AM +09:00
```

VALIDATE_CONVERSIONファンクションの追加

NEW IN 12.2

- CAST - 型変換が成功するか確認する
  - 成功は1。失敗は0。exprの評価結果がNULLの場合は1。exprの評価時にエラーが発生した場合は、エラーを返す。
  - 変換先はBINARY_DOUBLE, BINARY_FLOAT, DATE, INTERVAL DAY TO SECOND, INTERVAL YEAR TO MONTH, NUMBER, TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE

```
VALIDATE_CONVERSION(expr AS type_name [, fmt [, 'nlsparam' ] ])
```

- DATEへの変換の確認例

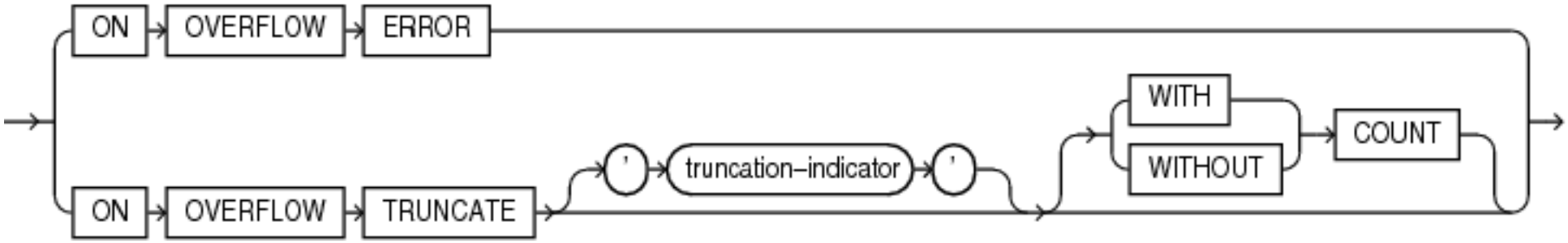
```
SELECT VALIDATE_CONVERSION('July 20, 1969, 20:18' AS DATE, 'Month dd, YYYY,
HH24:MI', 'NLS_DATE_LANGUAGE = American') FROM DUAL;

         V
----------
         1
```

LISTAGGファンクションの機能拡張(1/2)

NEW IN 12.2

OVERFLOW句の書式

- OVERFLOW句の追加



- オーバーフローする長さ
  - MAX_STRING_SIZE=EXTENDEDでは、VARCHAR2, RAWともに32767バイト
  - MAX_STRING_SIZE=STANDARDでは、VARCHAR2が4000バイト、RAWは2000バイト
  - ON OVERFLOW ERRORで文字数超過の場合” ORA-01489: result of string concatenation is too long”
- Trucation-indicatorはトランケートが発生した際に末尾に付加する文字列。デフォルトは”…”。
- WITH COUNTで、トランケートされた項目数の表示。WITHOUT COUNTでは表示しない。デフォルトは表示。

LISTAGGファンクションの機能拡張(2/2)

NEW IN 12.2

LISTAGGファンクションの実行例

- ALL_OBJECTSからOBJECT_NAMEのリストを取り出す。
  MAX_STRING_SIZE=STANDARDで、4000文字が上限

```
select listagg(object_name on overflow truncate '...' with count) within group
(order by object_name) as L from all_objects;

L
--------------------------------------------------------------------------------
.xdk_version_12.2.0.1.0_productionABSPATHABSPATHACCESSION_NUMBER_ORD_IMG_TACQUIS
ITION_DATETIME_ORD_IMG_TACQUISITION_DATE_ORD_IMG_TACQUISITION_NUMBER_ORD_IMG_TAC
[中略]
D_VIEWSALL_AW_OBJALL_AW_OBJALL_AW_PROPALL_AW_PROPALL_AW_PROP_NAMEALL_AW_PROP_NAMEALL
_AW_PSALL_AW_PSALL_BASE_TABLE_MVIEWSALL_BASE_TABLE_MVIEWSALL_CAPTUREALL_CAPTUREALL_C
APTURE_EXTRA_ATTRIBUTESALL_CAPTURE_EXTRA_ATTRIBUTESALL_CAPTURE_PARAMETERSALL_CAPTURE
_PARAMETERSALL_CAPTURE_PREPARED_DATABASEALL_CAPTURE_PREPARED_DATABASEALL_CAPTURE_PRE
PARED_SCHEMAS...(56116)
```

新しいコード・カバレッジ・ツールの提供(1/2)

NEW IN 12.2

DBMS_PLSQL_CODE_COVERAGEパッケージの追加

- DBMS_PLSQL_CODE_COVERAGEパッケージ
  - CREATE_COVERAGE_TABLES - コード・カバレッジを保存する表を作成
    - DBMSPCC_RUNS, DBMSPCC_UNITS, DBMSPCC_BLOCKS
  - START_COVERAGE - コード・カバレッジの取得開始
  - STOP_COVERAGE - コード・カバレッジの取得停止
  - 使い方はDBMS_PROFILERに類似(ただしパフォーマンス・データは対象としない)

新しいコード・カバレッジ・ツールの提供(2/2)

NEW IN 12.2

コード・カバレッジの収集例

```
dbms_plsql_code_coverage.start_coverage('test run');
exec coverage_test(101);
exec dbms_plsql_code_coverage.stop_coverage;
```

引数が101なので6行目が実行され、8行目は実行されない。

```
1: procedure coverage_test
2:  (ncreate  in number)
3: is
4: begin
5:   if (n > 100) then
6:     dbms_output.put_line('big number ' || to_char(n));
7:   else
8:     dbms_output.put_line('small number ' || to_char(n));
9:   end if;
10: end coverage_test;
```

BLOCK=2がLINE=6でCOVEREDが1(実行)、BLOCK=3がLINE=8でCOVERED=0(未実行)

DBMSPCC_BLOCKS

RUN_ID	OBJECT_ID	BLOCK	LINE	COL	COVERED	NOT_FEASIBLE
23	73892	1	1	1	1	0
23	73892	2	6	6	1	0
23	73892	3	8	6	0	0

PL/Scopeの拡張(1/2)

NEW IN 12.2

追加された機能

- PL/SQLユニットのコンパイル時に、識別子の情報に加えて、呼び出される静的および動的SQLの情報を取得する。
  - USER_STATEMENTS, ALL_STATEMENTS, DBA_STATEMENTS
  - ALTER SESSION SET PLSCOPE_SETTINGS = 'IDENTIFIERS:ALL, STATEMENTS:ALL';

PL/Scopeの拡張(2/2)

NEW IN 12.2

PL/ScopeによるSQL文の情報収集

```
alter session set PLSCOPE_SETTINGS = 'IDENTIFIERS:ALL, STATEMENTS:ALL';
alter procedure plscope_test compile;
```

PL/Scopeの設定を行い、コンパイルを実行

```
1 procedure plscope_test
2 is
3       n number;
4 begin
5       select count(*) into n from all_objects;
6       execute immediate 'select * from dual';
7 end plscope_test;
```

5行目は静的SQLなので、解析した情報を含む。6行目は動的SQLなので呼出し位置のみ。

```
select type, object_name, object_type, line, sql_id, text from user_statements
TYPE                OBJECT_NAME     OBJECT_TYPE   LINE  SQL_ID         TEXT
------------------- --------------- ------------ ----- ------------- --------------------------------
EXECUTE IMMEDIATE   PLSCOPE_TEST    PROCEDURE        6
SELECT              PLSCOPE_TEST    PROCEDURE        5 dr2v8xdg4w2ag SELECT COUNT(*) FROM ALL_OBJECTS
```

アジェンダ

1. SQL & PL/SQL
2. JSON
3. ユーティリティ(SQL*Plus, Data Pump, SQL*Loader)
4. グローバリゼーション・サポート
5. SQL & PL/SQL：追加
6. Java & JDBC

JavaとJDBCサポートの主な新機能

- JDK 8 とJDBC 4.2 のサポート
- 障害ノードの除外(Deprioritization)
- JDBCドライバによるFANイベントのサポート
- XAデータソースでのApplication Continuityのサポート
- JDBC Thinドライバでのネットワーク・データ圧縮
- SCN (System Change Number) サイズの拡大
- ALTER SESSIONでのサービス切り替え

JDK 8 とJDBC 4.2 のサポート

NEW IN 12.2

- %Large%メソッド群の追加
  - 更新行数などがint(Integer.MAX_VALUE = 2147483647)に収まらない場合に使用
    - executeLargeBatch(), executeLargeUpdate(), getLargeMaxRows(), getLargeUpdateCount(), setLargeMaxRows()
    - java.sql.Statementに追加
- setObject, updateObject, registerOutParameterメソッドへのSQLTypeパラメーター指定
  - 従来 int で指定していたオブジェクトを保存するカラムの型をSQLTypeにて指定
    - 従来: void setObject(int parameterIndex, java.lang.Object x, int targetSqlType) throws SQLException
    - 追加: void setObject(int parameterIndex, java.lang.Object x, SQLType targetSqlType) throws SQLException

障害ノードの除外(Deprioritization)

NEW IN 12.2

- ADDRESS_LISTに指定したノードへの接続に失敗した際、次回の接続時は失敗したノードへの接続を試みない。
- システム・プロパティ oracle.net.DOWN_HOSTS_TIMEOUT 経過後は、また試行対象となる。
- デフォルト値は10分。指定は秒単位。
- ホストAの接続に失敗すると、Bにつながり、oracle.net.DOWN_HOSTS_TIMEOUTが経過するまで、Aに接続はしない。
- TCP/IPレベルの接続障害。サービス名、ユーザ名などの問題は対象外。

URL指定の例

```
jdbc:oracle:thin:@
(DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=tcp)(HOST=A)(PORT=1521))
    (ADDRESS=(PROTOCOL=tcp)(HOST=B)(PORT=1521))
    (ADDRESS=(PROTOCOL=tcp)(HOST=C)(PORT=1521))
  )
  (CONNECT_DATA=(SERVICE_NAME=jpdb122))
)
```

JDBCドライバによるFAN(Fast Application Notification)イベントのサポート

NEW IN 12.2

- oracle.simplefan.FanUpEventListenerインターフェースの追加
  - handleEventにNodeUpEvent, ServiceUpEventが追加。
- JDBCドライバ単体でのFANイベントのサポート
  - UCP, Oracle Web Logicを使用している場合は不要。
  - サードパーティのコネクション・プールにてRACの高可用性に対応可能。
  - 計画停止では、コネクション・プール側でSafe Drain APIを使用して、安全に接続を閉じることができる瞬間(Draining-Point)を定義する。RAC側のサービス停止はDraining-Pointまで待つ。
  - 非計画停止についてはApplication Continuityの利用を検討する – Application Continuityについても12.2からJDBCドライバ単体でサポート。

XAデータソースでのApplication Continuityのサポート

NEW IN 12.2

- javax.sql.XADataSourceでのApplication Continuityのサポート
  - javax.sql.DataSourceと同様に動作。
- ローカル・トランザクションに限定
  - XAResourceによるトランザクションでは、開始時にACはディスエーブル。

JDBC Thinドライバでのネットワーク・データ圧縮

NEW IN 12.2

- Advanced Network CompressionのJDBC Thinドライバでの対応
  - 要Advanced Compressionオプション
  - Advanced Network CompressionについてWP参照
    http://www.oracle.com/technetwork/database/enterprise-edition/advancednetworkcompression-2141325.pdf
- 接続プロパティにて指定
  - "oracle.net.networkCompression" を "on" と指定 (= SQLNET.COMPRESSION)
  - “oracle.net.networkCompressionThreshold” はオプション指定 (= SQLNET.COMPRESSION_THRESHOLD)
  - COMPRESSION_LEVELS は high 限定

SCN (System Change Number) サイズの拡大

NEW IN 12.2

- 12.1までのJDBCドライバが扱うSCNのサイズは6バイト
- 12.2からは8バイトにサイズが拡大
  - より多くのトランザクション処理が可能になった。

ALTER SESSIONでのサービス切り替え

NEW IN 12.2

- 12.2から”ALTER SESSION”によりコンテナとサービスの切り替えが可能
  - 例: ALTER SESSION SET CONTAINER = JPDB122 SERVICE = JPDB122_P;
- 12.1までは以下のエラー
  - ORA-02248: ALTER SESSIONのオプションが無効です。

リファレンス

マニュアル・ドキュメント

- SQL言語リファレンス, 12cリリース2 (12.2)
  http://docs.oracle.com/cd/E82638_01/SQLRF/toc.htm
- PL/SQL言語リファレンス, 12cリリース2 (12.2)
  http://docs.oracle.com/cd/E82638_01/LNPLS/toc.htm
- JSON Developer's Guide, 12c Release 2 (12.2)
  http://docs.oracle.com/cd/E82638_01/ADJSN/toc.htm
- SQL*Plusユーザーズ・ガイドおよびリファレンス, 12cリリース2 (12.2)
  http://docs.oracle.com/cd/E82638_01/SQPUG/toc.htm

リファレンス

マニュアル・ドキュメント 続き

- ユーティリティ, 12cリリース2 (12.2)

  http://docs.oracle.com/cd/E82638_01/SUTIL/toc.htm

- Globalization Support Guide, 12c Release 2 (12.2)

  http://docs.oracle.com/cd/E82638_01/NLSPG/toc.htm

- JDBC Developer's Guide, 12c Release 2 (12.2)

  http://docs.oracle.com/cd/E82638_01/JJDBC/toc.htm

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Integrated Cloud

Applications & Platform Services


ORACLE®